COLPISCI I CERCHI
TEMPLATE DI PARTENZA

```
import pygame
import random
import math

# --- Inizializzazione ---
pygame.init()
LARGHEZZA, ALTEZZA = 600, 400
schermo = pygame.display.set_mode((LARGHEZZA, ALTEZZA))
pygame.display.set_caption("Colpisci i cerchi")
clock = pygame.time.Clock()

# --- Variabili ---
punteggio = 0

cerchi = []   # lista di cerchi: [x, y, colore, raggio]


# --- Funzioni ---
def disegna_cerchi(lista_cerchi):
    """Disegna tutti i cerchi"""
    # TODO: ciclo for per disegnare i cerchi con pygame.draw.circle


def muovi_cerchi(lista_cerchi, speed):
    """Aggiorna posizione verticale"""
    global punteggio
    # TODO: far scendere i cerchi
    # TODO: se un cerchio tocca il fondo, riportarlo in alto e togliere 1 punto


def aggiungi_cerchio(lista_cerchi):
    """Aggiunge un nuovo cerchio in alto con valori casuali"""
    # TODO: posizione x casuale tra 0 e LARGHEZZA, y=0
    # TODO: colore casuale
    # TODO: raggio fisso (es. 25)
    # TODO: aggiungerlo alla lista


def clic_su_cerchi(lista, pos_mouse):
    """Controlla se il clic è dentro un cerchio"""
    global punteggio
    # TODO: calcolare distanza mouse-centro
    # TODO: se distanza <= raggio → punteggio +1 e rimuovere il cerchio


# --- Ciclo principale ---
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.MOUSEBUTTONDOWN:
            clic_su_cerchi(cerchi, event.pos)
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                aggiungi_cerchio(cerchi)

    muovi_cerchi(cerchi, 2)

    schermo.fill((255,255,255))
    disegna_cerchi(cerchi)

    # Mostra punteggio
    font = pygame.font.SysFont(None, 40)
    testo = font.render(f"Punteggio: {punteggio}", True, (0,0,0))
    schermo.blit(testo, (10,10))

    pygame.display.flip()
    clock.tick(30)

pygame.quit()
```

Guida passo-passo: creare un gioco con Pygame

In questa guida vedremo come costruire un semplice gioco in cui l'obiettivo è **cliccare sui cerchi che cadono dallo schermo**. Partiamo dal template e arriviamo alla versione completa.

---

1. Introduzione a Pygame

Pygame è una libreria Python che permette di creare giochi e animazioni 2D. Fornisce strumenti per:

- Creare una finestra grafica
- Disegnare forme (cerchi, rettangoli, linee)
- Gestire eventi (clic del mouse, pressione di tasti, ...)
- Controllare il tempo (pygame.time.Clock) per gestire il frame rate

Il **frame rate** (o **FPS**, "frames per second") indica **quante immagini il gioco aggiorna in un secondo**. In pratica:

- Un frame = un'immagine sullo schermo
- Se il frame rate è basso (es. 5 FPS), il gioco appare **scattoso**
- Se il frame rate è alto (es. 60 FPS), il gioco appare **fluido**

In Pygame, il frame rate si controlla con:

```
clock = pygame.time.Clock()
clock.tick(30)  # limita il gioco a 30 FPS
```

Significa che il ciclo principale **non verrà eseguito più di 30 volte al secondo**.
Impostare un frame rate fisso aiuta a rendere il gioco coerente su computer diversi.

---

2. Lo spazio cartesiano in Pygame

Pygame usa un **sistema di coordinate** con l'origine (0,0) in alto a sinistra:

- **x** aumenta verso destra
- **y** aumenta verso il basso

Quindi il punto (300,200) si trova 300 pixel a destra e 200 pixel sotto l'angolo in alto a sinistra.

---

3. I TO-DO

Nel template sono presenti dei commenti # TODO. Questi indicano le parti del codice **che devono essere completate** in seguito:

- Disegnare i cerchi sullo schermo
- Aggiornare la posizione dei cerchi
- Creare nuovi cerchi con valori casuali
- Gestire il clic del mouse sui cerchi

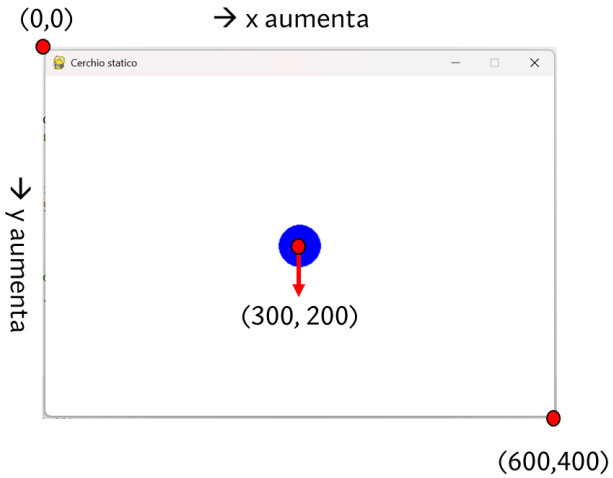
---

4. Variabili principali

- punteggio: tiene conto dei punti dell'utente
- cerchi: lista dei cerchi, ognuno rappresentato come [x, y, colore, raggio]

Esempio:

```
c1 = [300, 200, (0,0,255), 25]  # cerchio blu
cerchi = [c1]
```



in questo caso cerchi è una lista con un solo cerchio, il quale a sua volta è una lista che contiene 4 elementi di cui il terzo è una tupla che a differenza della lista è immutabile, i suoi valori non possono cambiare.

c1 è una **lista** con 4 elementi:

Indice	Contenuto	Tipo	Descrizione
0	300	int	Coordinata x
1	200	int	Coordinata y
2	(0, 0, 255)	tupla	Colore RGB (blu)
3	25	int	Raggio del cerchio

La **tupla (0,0,255)** è **immutabile**, cioè non puoi cambiare i suoi valori singolarmente, si può modificare solo l'intera tupla.

cerchi è una lista che contiene **un solo elemento**, cioè c1. Se aggiungi un altro cerchio, fai cerchi.append(c2) e ora la lista contiene due cerchi.

Nel nostro caso inizialmente abbiamo 3 cerchi:

```
c1 = [300, 200, (0, 0, 255), 25]
c2 = [400, 200, (255, 0, 0), 25]
c3 = [500, 200, (0, 255, 0), 25]
cerchi = [c1,c2,c3]
```

La lista cerchi è costituita da 3 elementi:

Indice	Contenuto	Tipo	Descrizione
0	c1	lista	Lista dati cerchio 1
1	c2	lista	Lista dati cerchio 2
2	c3	lista	Lista dati cerchio 3

5. Funzioni principali

a) Disegnare i cerchi

Definizione della funzione

```
def disegna_cerchi(lista_cerchi):
  """Disegna tutti i cerchi"""
  for c in lista_cerchi:
    pygame.draw.circle(schermo, c[2], (c[0], c[1]), c[3])
```

Come si definisce la funzione:

- **def**: parola chiave che serve per **definire una funzione**.
- **disegna_cerchi**: nome della funzione, usato per richiamarla in seguito.
- **lista_cerchi**: **parametro formale**. È un "segnaposto" che rappresenta la lista di cerchi che la funzione riceverà quando viene chiamata.

All'interno della funzione:

- Ciclo for c in lista_cerchi: scorre ogni cerchio nella lista
- pygame.draw.circle(schermo, c[2], (c[0], c[1]), c[3]) disegna il cerchio sullo schermo creato all'inizio del codice con usando come parametri:
  - schermo: finestra di Pygame
  - c[2]: colore
  - (c[0], c[1]): posizione (x, y)
  - c[3]: raggio

---

2. Invocazione della funzione

Quando chiamiamo la funzione:

```
disegna_cerchi(cerchi)
```

- cerchi è il **parametro attuale** (o argomento). È la **lista concreta** dei cerchi da disegnare che verranno copiati sul parametro formale lista_cerchi.
- *Python associa il parametro attuale cerchi al parametro formale lista_cerchi all'interno della funzione.*

ESEMPIO DI MATEMATICA

1. Definizione della funzione per il calcolo del quadrato di un numero $y = x^2$

```
def quadrato(x):
  """Restituisce il quadrato di x"""
  return x * x
```

- **def** → serve a definire la funzione
- **quadrato** → nome della funzione
- **x** → parametro formale: rappresenta il numero di cui vogliamo calcolare il quadrato
- **return** → restituisce il risultato della funzione

---

2. Invocazione della funzione

```
risultato = quadrato(5)
print(risultato)  # Output: 25
```

- 5 è il **parametro attuale** (argomento): il numero concreto che vogliamo usare
- x prende il valore 5 all'interno della funzione
- La funzione restituisce 5 * 5 = 25

b) Muovere i cerchi

```
def muovi_cerchi(lista_cerchi, speed):
  global punteggio
  for c in lista_cerchi:
    c[1] += speed  # sposta verso il basso
    if c[1] + c[3] > ALTEZZA:
      c[1] = 0  # torna in alto
      punteggio -= 1
      print("punteggio:", punteggio)
```

- c[1] è la coordinata y del centro dei vari cerchi, c[3] il raggio mentre ALTEZZA è la dimensione massima impostata per lo schermo lungo l'asse y.
- c[1] += speed → sposta il cerchio lungo l'asse y di una quantità pari a speed
  è la stessa cosa di: c[1] = c[1]+speed
- Se il cerchio supera il bordo inferiore, ricomincia dall'alto (y=0) e il punteggio diminuisce (punteggio -= 1 è la stessa cosa di punteggio = punteggio -1)
- Se proviamo a modificare una variabile globale senza dichiararla global, Python creerà una nuova variabile locale con lo stesso nome. La parola chiave global serve a dire "questa variabile non è locale, è quella definita all'esterno della funzione".

---

c) Aggiungere un cerchio casuale

```
def aggiungi_cerchio(lista_cerchi):
  x = random.randint(0, LARGHEZZA)
  y = 0
  colore = (random.randint(0,255), random.randint(0,255), random.randint(0,255))
  lista_cerchi.append([x, y, colore, 25])
```

- x casuale tra 0 e la larghezza dello schermo
- y=0 significa partire dall'alto
- colore casuale RGB
- Raggio fisso 25
- random è una libreria di Python che permette di **generare numeri casuali**. Prima di usarla, la importiamo con import random in cima al codice, poi possiamo usare varie funzioni, ad esempio: **random.randint(a, b)** → restituisce un numero **intero casuale tra a e b inclusi**.

---

d) Gestire il clic sui cerchi

```
def clic_su_cerchi(lista, pos_mouse):
  global punteggio
  for c in lista:
    distanza = math.sqrt((c[0]-pos_mouse[0])**2 + (c[1]-pos_mouse[1])**2)
    if distanza <= c[3]:
      punteggio += 1
      print("punteggio:", punteggio)
      lista.remove(c)
```

- Per ogni cerchio presente sullo schermo, calcola se il clic del mouse cade **all'interno**. Se sì, il giocatore guadagna un punto e il cerchio sparisce (viene rimosso dalla lista)
- Dichiarando global punteggio, diciamo alla funzione di **usare e modificare il punteggio globale**, non di crearne uno nuovo locale.
- print serve solo per vedere il punteggio nel terminale
- Calcoliamo la **distanza** tra il clic e il centro del cerchio
- lista: è la lista di cerchi presenti sullo schermo.
- pos_mouse: è la posizione del clic del mouse, una tupla (x, y) che indica dove l'utente ha cliccato (pos_mouse[0] corrisponde alla x del mouse).
- Usiamo il teorema di Pitagora per calcolare la distanza tra il centro del cerchio (c[0], c[1]) e il punto cliccato dal mouse (pos_mouse[0], pos_mouse[1]).
- Se la distanza è minore o uguale al raggio, il cerchio viene rimosso e il punteggio aumenta

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

---

6. Ciclo principale

```
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
    elif event.type == pygame.MOUSEBUTTONDOWN:
      clic_su_cerchi(cerchi, event.pos)
    elif event.type == pygame.KEYDOWN:
      if event.key == pygame.K_SPACE:
        aggiungi_cerchio(cerchi)

  muovi_cerchi(cerchi, 1)
  schermo.fill((255,255,255))
  disegna_cerchi(cerchi)

  font = pygame.font.SysFont(None, 40)
  testo = font.render(f"Punteggio: {punteggio}", True, (0,0,0))
  schermo.blit(testo, (10,10))

  pygame.display.flip()
  clock.tick(30)
```

- Il ciclo while running: mantiene il gioco attivo finché running=True
- pygame.event.get() raccoglie gli **eventi** (clic, tasti, chiusura finestra)
- schermo.fill pulisce lo sfondo
- pygame.display.flip() aggiorna lo schermo
- clock.tick(30) mantiene 30 FPS (frame al secondo)

Il ciclo principale legge continuamente gli eventi generati dall'utente. In base al tipo di evento, il gioco reagisce:

- Chiusura finestra (QUIT) → termina il ciclo e chiude il gioco.
- Clic del mouse (MOUSEBUTTONDOWN) → controlla se il clic è dentro un cerchio e aggiorna il punteggio.
- Premuta di un tasto (KEYDOWN) → se è spazio, aggiunge un nuovo cerchio in cima allo schermo.

7. Risultato finale

Alla fine, implementando tutte le funzioni, il gioco sarà completo.
L'utente può:

- Premere **spazio** per aggiungere un nuovo cerchio
- Cliccare sui cerchi per guadagnare punti
- Vedere il punteggio aggiornarsi in tempo reale

Riepilogo

1. La libreria Pygame

- Pygame è una libreria Python pensata per creare **giochi e animazioni 2D**.
- Fornisce strumenti per:
  - Creare finestre grafiche.
  - Disegnare forme (cerchi, rettangoli, linee).
  - Gestire eventi dell'utente (clic del mouse, tasti premuti, chiusura finestra).
  - Controllare il tempo e il frame rate.

---

2. Lo spazio cartesiano in Pygame

- L'origine (0,0) si trova **in alto a sinistra** dello schermo.
- L'asse **x** aumenta verso destra.
- L'asse **y** aumenta verso il basso.
- Tutte le posizioni e i movimenti degli oggetti si basano su questo sistema di coordinate.

---

3. Liste e tuple

- Ogni cerchio è rappresentato da una **lista di quattro elementi**: posizione x, posizione y, colore, raggio.
- Il colore è una **tupla RGB**.
  - La tupla è **immutabile**: i valori al suo interno non possono essere modificati singolarmente.
- La lista dei cerchi contiene tutte le liste dei singoli cerchi ed è **mutabile**, quindi possiamo aggiungere o rimuovere cerchi.

---

4. Parametri e funzioni

- Una **funzione** è un blocco di codice che svolge un compito specifico.
- I **parametri formali** sono variabili definite nella funzione come segnaposto.
- I **parametri attuali** (argomenti) sono i valori concreti passati alla funzione quando viene chiamata.

---

5. Ciclo principale e gestione eventi

- Il gioco gira dentro un **ciclo principale** che si ripete continuamente.
- Durante ogni ciclo:
    - Si leggono gli **eventi dell'utente** (clic del mouse, tasti premuti, chiusura finestra).
    - Si aggiornano le posizioni degli oggetti.
    - Si ridisegna lo schermo.
- Questo ciclo permette di avere un gioco **interattivo e dinamico**.

---

6. Frame rate (FPS)

- Il frame rate indica **quante volte al secondo lo schermo viene aggiornato**.
- Un frame corrisponde a un'immagine visualizzata.
- Limitare il frame rate rende il gioco **fluido** e coerente su diversi computer.
- La velocità degli oggetti sullo schermo dipende dal frame rate e dalla velocità impostata.

---

7. Variabili globali

- Le variabili definite fuori dalle funzioni sono **globali**.
- Per modificare una variabile globale dentro una funzione, si usa la parola chiave **global**.
- Questo permette, ad esempio, di aggiornare il punteggio reale del gioco dentro le funzioni.

---

8. Generazione casuale

- La libreria **random** permette di generare numeri casuali.
- In un gioco serve per creare **posizioni e colori variabili**, rendendo il gioco più imprevedibile e divertente.
- Ad esempio, la posizione orizzontale dei cerchi o il loro colore possono essere scelti casualmente.

---

9. Calcolo delle distanze

- Per rilevare se un clic del mouse cade dentro un cerchio si usa la **distanza tra due punti**.
- Si applica il **teorema di Pitagora** per calcolare la distanza tra il centro del cerchio e il punto cliccato.
- Se la distanza è minore o uguale al raggio del cerchio, significa che il clic è avvenuto **all'interno** del cerchio.

---

10. Comportamento del gioco

- I cerchi **cadono verso il basso** e se raggiungono il fondo ritornano in alto.
- Se il giocatore clicca sul cerchio, il cerchio **scompare** e il punteggio **aumenta**.
- Se un cerchio raggiunge il fondo senza essere cliccato, il punteggio **diminuisce**.
- Premendo un tasto specifico (ad esempio spazio), si può **aggiungere un nuovo cerchio** in alto con colore e posizione casuale.

SOLUZIONE

```
import pygame
import random
import math

# --- Inizializzazione ---
pygame.init()
LARGHEZZA, ALTEZZA = 600, 400
schermo = pygame.display.set_mode((LARGHEZZA, ALTEZZA))
pygame.display.set_caption("Colpisci i cerchi")
clock = pygame.time.Clock()

# Cerchio: [x, y, colore, raggio]
c1 = [300, 200, (0, 0, 255), 25]
c2 = [400, 200, (255, 0, 0), 25]
c3 = [500, 200, (0, 255, 0), 25]
cerchi = [c1,c2,c3]

punteggio = 0

# --- Funzioni ---
def disegna_cerchi(lista_cerchi):
    """Disegna un cerchio sullo schermo"""
    for c in lista_cerchi:
        pygame.draw.circle(schermo, c[2], (c[0], c[1]), c[3])

def muovi_cerchi(lista_cerchi, speed):
    global punteggio
    for c in lista_cerchi:
        c[1] += speed  # y = y + velocità
        if c[1] + c[3] > ALTEZZA:  # se tocca il fondo, ricomincia dall'alto
            c[1] = 0
            punteggio -= 1
            print("punteggio: ",punteggio)

def aggiungi_cerchio(lista_cerchi):
    """Aggiunge un nuovo cerchio in cima"""
    x = random.randint(0, LARGHEZZA)
    y = 0
    colore = (random.randint(0,255), random.randint(0,255), random.randint(0,255))
    lista_cerchi.append([x, y, colore, 25])

def clic_su_cerchi(lista, pos_mouse):
    """Controlla se il clic è dentro un cerchio"""
    global punteggio
    for c in lista:
        distanza = math.sqrt((c[0]-pos_mouse[0])**2 + (c[1]-pos_mouse[1])**2)
        if distanza <= c[3]:
            punteggio += 1
            print("punteggio: ",punteggio)
            lista.remove(c)

# --- Ciclo principale ---
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.MOUSEBUTTONDOWN:
            clic_su_cerchi(cerchi, event.pos)
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE:
                aggiungi_cerchio(cerchi)

    muovi_cerchi(cerchi, 1)
    schermo.fill((255,255,255))  # sfondo bianco
    disegna_cerchi(cerchi)     # disegna il cerchio

    # Mostra punteggio
    font = pygame.font.SysFont(None, 40)
    testo = font.render(f"Punteggio: {punteggio}", True, (0,0,0))
    schermo.blit(testo, (10,10))

    pygame.display.flip()
    clock.tick(30)

pygame.quit()
```